Why Structured Parallel Programming Matters

Why Structured Parallel Programming Matters

Murray Cole

School of Informatics, University of Edinburgh,
King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ, Scotland
mic@inf.ed.ac.uk

Abstract. Simple parallel programming frameworks such as Pthreads, or the six function core of MPI, are universal in the sense that they support the expression of arbitrarily complex patterns of computation and interaction between concurrent activities. Pragmatically, their descriptive power is constrained only by the programmer's creativity and capacity for attention to detail.

Meanwhile, as our understanding of the structure of parallel algorithms develops, it has become clear that many parallel applications can be characterized and classified by their adherence to one or more of a number of generic patterns. For example, many diverse applications share the underlying control and data flow of the pipeline paradigm, whether expressed in terms of message passing, or by constrained access to shared data.

A number of research programs, using terms such as *skeleton*, *template*, *archetype* and *pattern*, have sought to exploit this phenomenon by allowing the programmer to explicitly express such meta-knowledge in the program source, through the use of new libraries, annotations and control constructs, rather than leaving it implicit in the interplay of more primitive universal mechanisms.

While early work stressed productivity and portability (the programmer is no longer required to repeatedly "reinvent the wheel") we argue that the true significance of this approach lies in the capture of complex algorithmic knowledge which would be impossible to determine by static examination of an equivalent unstructured source. This enables developments in a number of areas. With respect to *low-level performance*, it allows the run-time system, library code or compiler to make clever optimizations based on detailed foreknowledge of the evolving computation. With respect to *high-level performance*, it enables a methodology of improvement through powerful restructuring transformations. Similarly, with respect to *program correctness*, it allows arguments to be pursued at a much coarser, more tractable grain than would otherwise be possible.

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, p. 37, 2004.